二分图匹配

2019年8月5日　　22:43

-
- 二分图的匹配

一. 二分图/二部图/偶图：将无向图的N个节点划分进两个非空集合，集合内点之间无边

1. Thm：二分图 iff 没有奇环（含奇数条边的环）
2. 染色法判定二分图：从任意点出发，dfs给相邻点染异色，若相邻点已有颜色且与目标染色不同，说明不是二分图，注意此时return0需要能被传递到起点
3. `int clr[MN]; //clr[x] = x染成颜色1或颜色2或暂时是颜色0，即尚未被染色`

```
bool cdfs(int x,int c=1){ //将x染色成c，返回能否成功染色成二分图
     clr[x] =c;
     for(int i= fst[x]; i; i= nxt[i]){
          int y= to[i];
          if(!clr[y]){
               if(!cdfs(y, 3-c))
                    return 0;
          }else
               if(clr[y]==c)
                    return 0;}
     return 1; }
```

4. `for__(x,1,n)`

```
     if(!clr[x])
          cdfs(x);
```

5. 例题：关押罪犯（P1525）
   i. 题意：将n个罪犯关进两个监狱，求单个监狱内最大矛盾值的最小值
   ii. 二分判断各mid值能否作为最大矛盾值
   iii. 即若边权<=mid，就忽视他们，若边权>mid，则把他们分进二分图，若能成功分进二分图，说明两个监狱内都只有<=mid的矛盾

二. 二分图最大匹配/边独立集

1. 匹配：任两边都无公共端点的边的集合
   i. 匹配边：匹配内的边；匹配点/饱和点：匹配边的端点
   ii. 最大匹配：图中包含边数最多的匹配
2. 增广路/交错路：匹配边和非匹配边交替出现的、连通非匹配点的路径
   i. 引：增广路上第奇数条边是非匹配边；第偶数条边是匹配边
      1) 记第偶数条边组成的集合为匹配S，称该增广路为S的增广路
      2) 推：增广路总长度一定是奇数
   ii. Thm：增广路上第奇数条边的集合也是匹配
      1) 应用：将第偶数条匹配边改成非匹配边，再把第奇数条非匹配边改成匹配边，会得到边数多1的新匹配
      2) 推：匹配S是最大匹配 iff 原图中不存在S的增广路
3. 匈牙利算法/增广路贪心算法
   i. 初值设所有边都不是匹配边
   ii. 找非匹配边加入增广路，并对增广路径上匹配边状态取反
   iii. 直至找不到增广路时即得到了最大匹配

4. dfs实现匈牙利算法
    i. 对非饱和点x，遍历每个相邻点y：
        1) 若y也是非饱和点，则边(x,y)本身就是一条增广路径，取反后，y就会匹配上x，再return 1，表示x出发能找到增广路
        2) 若y是饱和点，y匹配上了x0，则对x0遍历其相邻点，若能找到别的匹配点（集合）y'，说明从x出发能形成x~y~x0~y'的增广路
    ii. 主调函数中遍历每个点，确认能找到新增广路时说明匹配大小++
    iii. 每次有新结点尝试加入匹配，只会更改当前已匹配结点的匹配对象，不会使当前匹配中的结点退出匹配，保证了每次新一起点的dfs不会使匹配大小减少
    iv. N个点都要作为一次dfs起点才能保证不漏，运气差时找增广路需要遍历几乎全部M条边，时间复杂度O（NM）

```
int vstl[MN],vstr[MN];
int mchl[MN],mchr[MN];     //左部点匹配到的右部点，右部点匹配的左部点
bool hdfs(int x){   //返回从左部点x出发能否找到新增广路
     if(vstl[x]) return 0;
     vstl[x] =1;
     for(int i= fst[x]; i; i= nxt[i]){
          //if(!vstr[ y= to[i] ]){
          //     vstr[y] =1;
          int y= to[i];
               if(!mchr[y] || hdfs(mchr[y])){
                    mchr[y] =x;  //决定让(y,x)从非匹配边转成匹配边
                    return 1;}}
     return 0;}
for__(x,1,n){
     for__(l,1,n) vstl[l] =0;
     if(hdfs(x)) ++ans;}
```

5. 二分图匹配模型
    i. 0要素：点集合内只有0条边
    ii. 1要素：每个点只有一条关联边
6. 例题：超级英雄（P2319）
    i. m道题分别用两种锦囊可以解，求最多做出连续前几道题，及用的锦囊
    ii. 依次对前几道题跑匹配即可，不过不一定是求最大匹配，碰到第一个搜不出匹配锦囊的题（即第一个dfs返回0的点）就退出匹配过程
7. 例题：棋盘覆盖（ACWING372）
    i. 在黑白棋盘上放1x2的牌，部分格子禁止放牌，求最多能放多少
    ii. 0要素：同色格之间无边；1要素：各格只能放一牌
    iii. 把n*n个二维格子离散化成n*n个结点，向非禁区连通点连边
    iv. 只从黑色或白色格出发的匈牙利算法，即可得到边数=牌数
    v. n*n个点，各能关联最多4条边，复杂度O（$4n^4$）

三. 其他匹配

1. 完备匹配：二分图中，左右两部都只有N个结点，全最大匹配恰有N条边，称该二分图有完备匹配
    i. 完美匹配：每个结点都可以是饱和结点的图，一定是完备匹配
2. 多重匹配：每个点可与规定不超过xi条边相关联的广义匹配
    i. 拆点法：把结点i视作xi个结点，再求最大匹配

ii. 若只有一部是多重匹配，则可在该部对每个结点执行xi次匈牙利dfs

iii. 网络最大流解法：令所有左结点到所有右结点的流容量为无穷，令源点到各左结点的容量为各左节点匹配上限，令所有右结点到汇点的容量~

四. 二分图带权最大匹配

1. 带权最大匹配/最优匹配：边带权图的最大匹配中，权值和也最大的匹配
   i. KM算法：在稠密图效率高，但只适用于NN点的完备匹配
   ii. 费用流解法
2. 交错树：匈牙利算法匹配失败时dfs过程中访问过的点和边组成的树
   i. 根和叶子同属一部，设为左部
   ii. 树上奇数层的边都是非匹配边，偶数层的都是匹配边（左部根结点出发遍历各边为第一层，关联到的都是右部点，从右部点的匹配点出发深搜时，相当于将匹配边放在偶数层，而这些左部点再遍历时又会尝试将新的非匹配边放入奇数层，即：**根以外的左部点都是通过mch数组访问**的，而所有**右部点都是从左部点沿非匹配边访问到的**）
3. 顶点标记值/顶标：二分图中各顶点的权值，要求边权<=关联点顶标之和
   i. 相等子图：二分图中所有匹配点权和==匹配边权的边组成的子图
   ii. 引：相等子图中完备匹配的边权和==顶标之和
   iii. Thm：相等子图中有完备匹配 iff 该匹配是带权最大匹配
4. KM算法：给二分图任意配满足要求的顶标，再尝试缩小顶标并求完备匹配
   i. 首先要确保结点能明显的分成左右两部，设出发点都是左部
   ii. 右部顶标初值0，左部顶标初值为关联边最大权，delta每轮初值无穷大
   iii. 在匈牙利dfs中修改全局变量delta的取值，令左部顶标减少，右部增加
5. delta的用法：dfs中不断减小，使尽可能多的边能进入相同子图，在dfs失败后微调交错树中的顶标（设交错树结点集合T，左部L，右部R）
   i. 引：根节点外的L点都是R点沿匹配边访问的，这个没法改；R点都是L点沿非匹配边访问到的，R点都是L点沿非匹配边访问到的，L点顶标减小后可访问更多不属于T的R点
   ii. 这是以为你当一对L点和R点都属于T时，L-=delta和R+=delta使该匹配边任属于相等子图；当L点属于T而R点不属于T时，L点顶标减小而R点顶标尚未增大，任为0，可加入T
   iii. 为了保证顶标不等式恒成立，每次delta取关于可加入T的R点的最小值
   iv. 当原图存在完备匹配时可以保证每次都能往相等子图加入新边直至完备
6.
```
int mch[MN]; //mch[x]=x对应的匹配点
int vl[MN],vr[MN];  //左右部各结点是否在交错树中，每次hdfs前清空
int dl[MN],dr[MN];  //左右部各结点的顶标
int vlr[MN][MN];    //左部点到右部点的边权
int delta,dt[MN];   //顶标修改值及其备选值
```
7.
```
bool hdfs(int x){   //返回x出发能否找到新增广路
    vl[x] = 1;   //为了知道是否需要改顶标，左部结点也许判断遍历标记
    int y; //注意这个y不能用全局变量！否则会在回溯时变成子结点！
    for(int i= fst[x]; i; i= nxt[i])
```

```
            if(!vr[ y= to[i] ])
                if(dl[x] +dr[y] -val[i] ==0){//确认是相等子图的边
                    vr[y] =1;    //确认该点是想改好delta后加入的点
                    if(!mch[y] || hdfs(mch[y])){
                        mch[y] =x;   //决定选(y,x)加入匹配边
                        return 1;
                    }
                }else //先不急着更新delta
                    dt[y]= min(dt[y], dl[x] +dr[y] - dst[x][y]);
                return 0;
    }
8.  ll KM(){
        for__(x,1,n){
            dr[x]=0;
            dl[x]=-(1<<30);     //初值负无穷，再改为最大邻边
            for(int i= fst[x]; i; i= nxt[i])
                dl[x]= max(dl[x], val[i]);
        }
        for__(x,1,n) //让每个左部结点x都找到对应匹配
            for(;;){
                ms(vl,0);
                ms(vr,0);
                if(hdfs(x))  //已经加入新边了，该找下一个左部点了
                    break;
                delta = 1<<30;      //目标是最小相等子图所需边
                for__(y,1,n)
                    if(!vr[y])   //确认肯定不在交错树中，再更新delta
                        delta= min(delta, dt[y]);
                for__(t,1,n){
                    if(vl[t])
                        dl[t] -=delta;
                    if(vr[t])
                        dr[t] +=delta;
                }
            }
        ll ans =0;
        for__(x,1,n)
            ans+= vlr[mch[x]][x];
        return ans;
    }
```

9. 例题：Ants黑白点连不相交边最小值（POJ3565）（UVA1411）
    i. 输入数据保证有解。由三角形两边之和大于第三边可知交叉线段可转换成对顶三角的两条底边，于是最小匹配的总边长必满足不相交
    ii. 求最小匹配的方法是把边权全改成负数，再求最大匹配